# On Programming of Network Services

Ascan F. Morlang, Dr. Ina Schieferdecker
GMD FOKUS, Kaiserin-Augusta-Allee 31, D-10589 Berlin, Germany
Tel. +49 30 343 - 7138, Fax. -8138
{morlang,schieferdecker}@fokus.gmd.de

## Abstract

*This paper introduces our ongoing development of a network node architecture, which meets the requirements of dynamic run-time installation and configuration of network services. The objective is to provide a network service run-time system, which is independent from installed services and suitable for effective deployment of existing, new and future services. By developing network service specific resource abstractions, the architecture will be more flexible than existing approaches.*

## 1. Introduction

The success of the Internet has encouraged many proposals for novel applications and new usage and business scenarios with the need for modifications inside the network. Starting with [1] most proposals made towards a flexible networking environment focused on the network stack inside the communication end host. Due to the different environment in an intermediate node, the results of the former approaches cannot be directly transferred, and hence new concepts have to be developed.

## 2. Network services and flows

The traditional process model was developed to support continuous, independent activities inside one computing system. In contrast, network services are more or less distributed functions applied on incoming packets with few or non continuous elements. Therefore our approach was to develop a more suitable abstraction, called service abstraction.

Services are modelled as an aggregation of service objects, which may consist of service objects themselves, which in the end consist of “Basic Service Objects” (*BSO*s).

The *BSOs* has zero or more input and output interfaces (data-path interfaces). Additionally, they provide some basic computational methods (computational interface) usable on themselves, e.g. for life-cycle management. Beside the basic methods, each *BSO*-class may implement specific operations, used e.g. for parameter transfer or status control. The computational interface is only accessible from the inside of the node, but can be made public to connected hosts through *BSO*s, which implement configuration or management services.

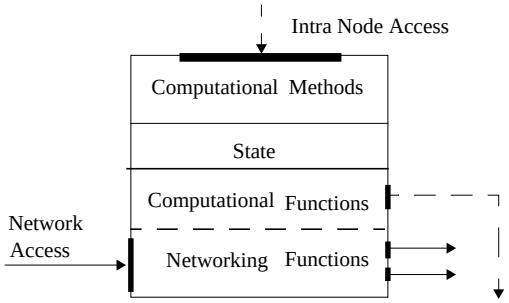


**Figure 1: Basic Service Object**

The intra-node interfaces can be used for run-time binding to already running service instances, like a VPN-routing service. The service binding interface has to be offered by the underlying run-time system, which exports locally installed objects. Objects bound to other services may become part of those and act on behalf of the caller’s resources or stay independent as separate services with own resource pools.

To allow an interconnection of service objects, the need to distinguish between different flows is obvious. Since the definition of the term flow and the applied distinction methods differ depending on the associated services, our service run-time system does not incorporate static flow distinction methods and leaves the implementation to the service programmer. The system identifies a flow uniquely by its source and sink, as a path which the network data takes inside the node. Source and Sink may be any computational objects, e.g. service objects or network interfaces allowing an stream to be separated into flows depending on the input network interface.

Flow distinction based on protocol fields can be implemented by service objects, which classify the input by the supplied filter parameters. Depending on the expected input or the resulting table sizes, different filter algorithms can be applied to divide global look-up tables into smaller ones.

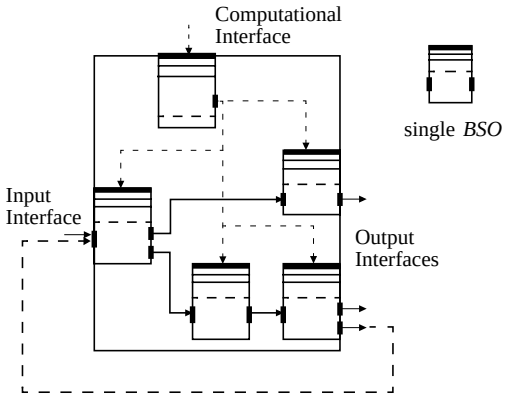


**Figure 2: Service Instance**

### 2.1. Resource access and protection

The node design concentrates on intra-node security to guarantee that any misbehaving service can be prevented from affecting the overall performance substantially. The problem of secure service access is left to the service implementation.

Traditional operating systems are based on the user abstraction for resource access and the process abstractions for resource protection. Since the user abstraction has no natural meaning to an intermediate network node, an Access Control List based approach is not optimal as a resource management concept. Therefore, a capability based approach is more promising. Each resource granted to an entity is represented by an associated object, which defines the actual rights to access or modify this resource. Following [2], an extended capability model including consumable resources is used to control the total system performance and to enable fault isolation mechanisms.

Capabilities are granted at service creation time by the creating service and may be shared during service execution between services. During boot-time, the first service holds all available capabilities and launches the initial service configuration.

Another important difference to traditional operating systems is the need for dynamic resource allocation during service execution. Since networks introduce real-time constraints, a service must be supplied with the required resources to complete computation within the given deadline. On the other hand, unused resources should be consumed by other services, to improve overall performance.

## 3. Related Work

The Active Networking Node project [3] aims to develop an architecture for the dynamic deployment of protocols at a high processing speed. It is based on a BDS-operating system with an *IPvX* environment and defines code modules for the dynamic extension of the kernel network stack.

The Scout [4] project, a flexible networking system, is a promising target for the implementation of the proposed architecture. Scout eliminates the process abstraction in favour to a communication path oriented scheduling, which is similar to the presented service oriented model. Concentrating on the communication end host, the optimization of the bi-directional paths was one main developing goal, therefore only a limited protection model has been implemented.

## 4. Conclusions

The trade-off between performance and flexibility limits possible application areas for the proposed service architecture. The node has to transfer the packets back and forth from the input interfaces to the main memory, perform all computation and manage the internal resources. Therefore, it is impossible to compete in the backbone with legacy routers without specialized hardware.

The access nodes at network boundaries are a more appealing field of application, as the packet rates are not so high and flexibility is crucial. Examples are the customer/provider boundary and the transition from one transport medium to another, for example at a base station connecting a wireless LAN with the wired infrastructure.

The presented service architecture opens novel possibilities for network service design and deployment. The deployment of architecture will shorten response times on demands for changes inside protocols and services and make individual networking solutions possible.

## References

[1] Dennis M. Ritchie, “A Stream Input-Output Interface”, AT&T Bell Laboratories Technical Journal 63, No. 8, October 1984, pp. 1897-1910

[2] J.S. Shapiro, J.M. Smith and D.J. Faber, “Eros: A Capability System”, Tech. Report, University of Pennsylvania, June 1997

[3] Dan Decasper, Guru Parulkar, Bernhard Plattner, “A Scalable, High Performance Active Network Node”, Proceedings of INFOCOM’98, April 1998

[4] L. L. Peterson, David Mosberger, et. al., “Scout: A Communications-Oriented Operating System”, Technical Report, University of Arizona, Tuscon, June 1994